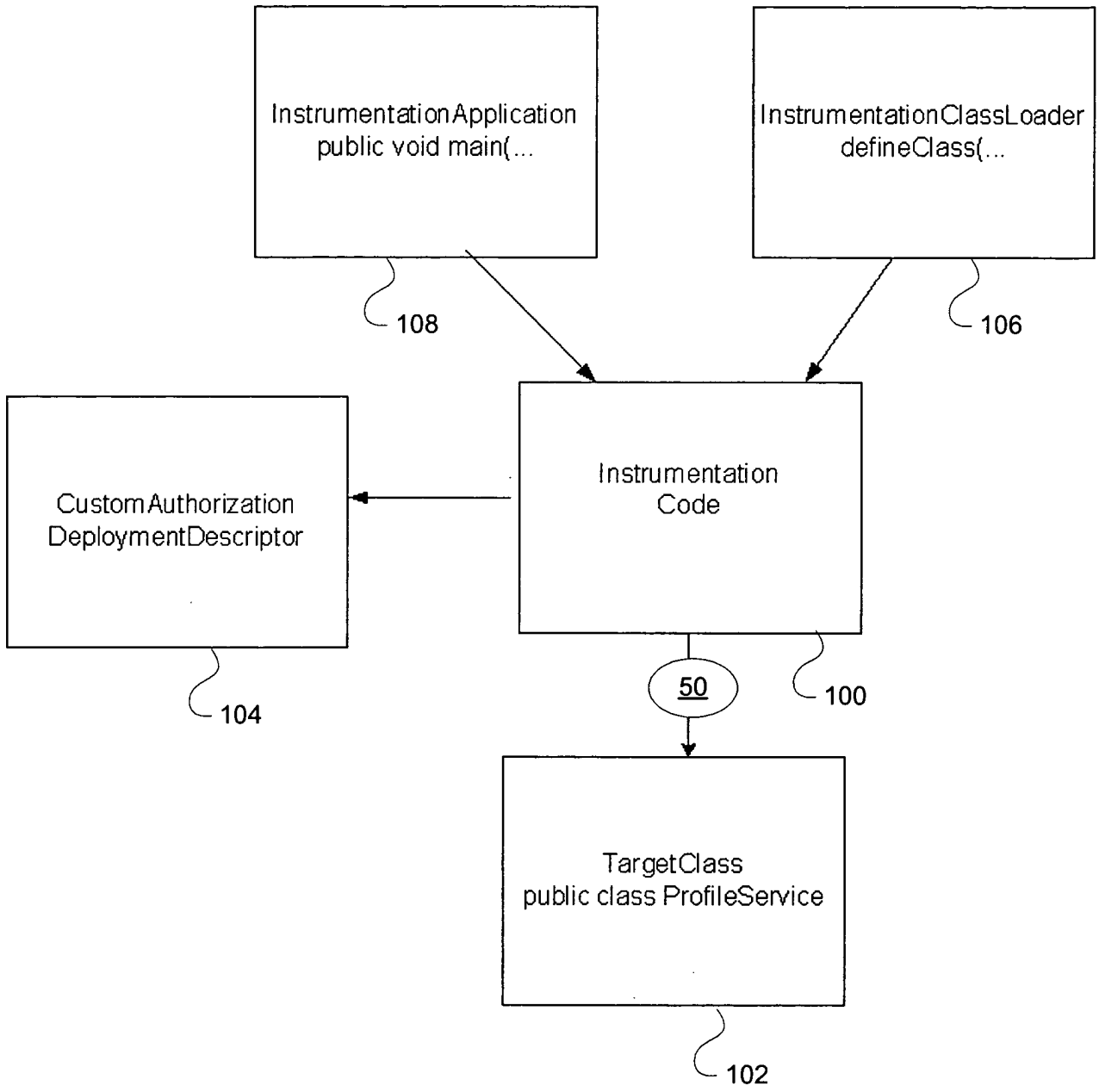
Extended Authorization Framework
InstrumentationApplication
public void main(...
InstrumentationClassLoader
defineClass(...
108
106
CustomAuthorization
DeploymentDescriptor
Instrumentation
Code
104
50
100
TargetClass
public class ProfileService
102

Fig. 1

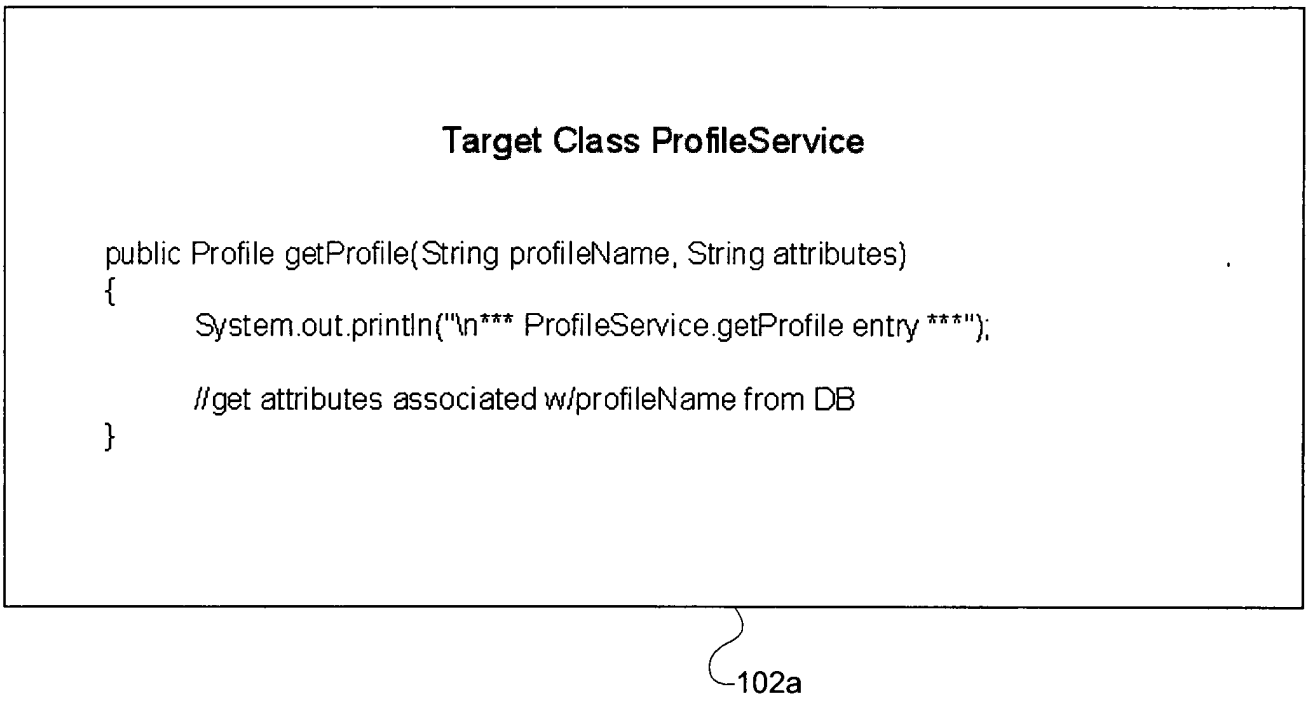
Target Class ProfileService
public Profile getProfile(String profileName, String attributes)
{
System.out.println("\n*** ProfileService.getProfile entry ***");
//get attributes associated w/profileName from DB
}
102a

Fig. 2

Custom Authorization Deployment Descriptor

```
...
<authorizationTarget class="ProfileService">

        <subjectFactory class="ThreadSubjectFactory"/>

        <method name="getProfile">
                <requiredPermission class ="com.ibm.resource.security.auth.ProfilePermission">
                        <property name="name" value="UserProfile" />
                        <property name="actions" value="Read" />
                        <property name="attributes" value="attributes" scope="local" />
                </requiredPermission>

                <privilegedAction class ="ReadAction">
                        <property name="attributes" value="attributes" scope="local" />
                </privilegedAction>

        </method>


</authorizationTarget>
...
```

104a

Fig. 3

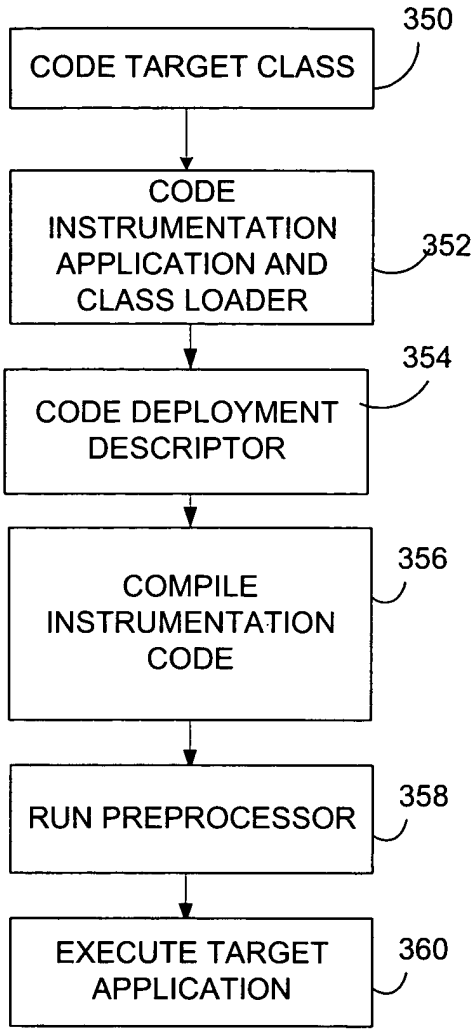
CODE TARGET CLASS
350
CODE INSTRUMENTATION APPLICATION AND CLASS LOADER
352
CODE DEPLOYMENT DESCRIPTOR
354
COMPILE INSTRUMENTATION CODE
356
RUN PREPROCESSOR
358
EXECUTE TARGET APPLICATION
360

Fig. 4

Target Class ProfileService — Instrumented Code

```
public Profile getProfile(String profileName, String attributes)
{

        System.out.println("\n*** ProfileService.getProfile entry ***");

/* JAVA LANGUAGE EQUIVALENT OF INSTRUMENTATION
(It's actually inserted into the target as byte code)
*/
SubjectFactory subjectFactory= new ThreadSubjectFactory();
Subject user =subjectFactory.getSubject();

if(user != null  &&  (System.getProperty("custom_authorization") != null))
{

PermissionFactory permissionFactory = (new DefaultPermissionFactory()).getFactory();

permissionFactory.setProperty("RequiredPermission","com.ibm.resource.security.auth.ProfilePermission");
permissionFactory.setProperty("RequiredPermission.name", "UserProfile");
permissionFactory.setProperty("RequiredPermission.actions", "read");
permissionFactory.setProperty("RequiredPermission.attributes", attributes);


PrivilegedActionFactory privilegedActionFactory =  (new DefaultPrivilegedActionFactory()).getFactory();
privilegedActionFactory.setProperty("PermissionFactory", permissionFactory);
privilegedActionFactory.setProperty("PrivilegedAction", "ReadAction");
privilegedActionFactory.setProperty("attributes", attributes);


Subject.doAsPrivileged(user, privilegedActionFactory.getPrivilegedAction(), null);

}
/*END INSTRUMENTATION */


        //get attributes associated w/profileName from DB
}
```

102b

Fig. 5

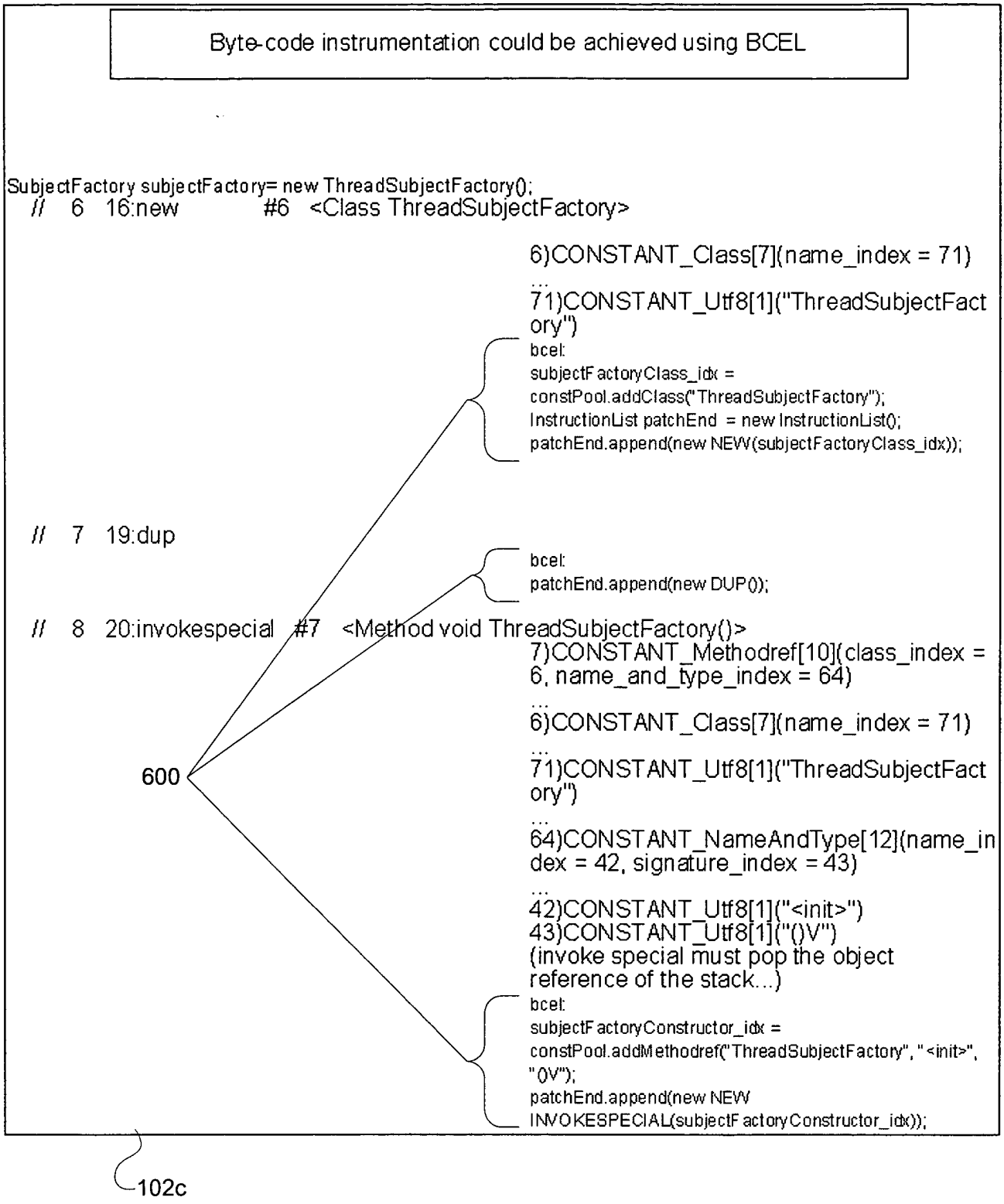
Byte-code instrumentation could be achieved using BCEL
SubjectFactory subjectFactory= new ThreadSubjectFactory();
// 6 16:new #6 <Class ThreadSubjectFactory>
6)CONSTANT_Class[7](name_index = 71)
...
71)CONSTANT_Utf8[1]("ThreadSubjectFact ory")
bcel:
subjectFactoryClass_idx = constPool.addClass("ThreadSubjectFactory");
InstructionList patchEnd = new InstructionList();
patchEnd.append(new NEW(subjectFactoryClass_idx));
// 7 19:dup
bcel:
patchEnd.append(new DUP());
// 8 20:invokespecial #7 <Method void ThreadSubjectFactory()>
7)CONSTANT_Methodref[10](class_index = 6, name_and_type_index = 64)
...
6)CONSTANT_Class[7](name_index = 71)
...
71)CONSTANT_Utf8[1]("ThreadSubjectFact ory")
...
64)CONSTANT_NameAndType[12](name_in dex = 42, signature_index = 43)
...
42)CONSTANT_Utf8[1]("<init>")
43)CONSTANT_Utf8[1]("()V")
(invoke special must pop the object reference of the stack...)
bcel:
subjectFactoryConstructor_idx = constPool.addMethodref("ThreadSubjectFactory", "<init>", "()V");
patchEnd.append(new NEW INVOKESPECIAL(subjectFactoryConstructor_idx));
600
102c

Fig. 6